

**Django FrameWork – Ambiente de desenvolvimento**

**Gabriel Rosa, Gustavo Garcia, Pierre Fenner**

**Universidade Franciscana(UFN) - Santa Maria – RS**

**1.Conhecendo**

Django é Web framework de alto nível do Python que proporciona ao programador um desenvolvimento prático, limpo de websites seguros e com fácil manutenção. Django se responsabiliza pela maior parte “chata” do desenvolvimento de uma aplicação, deste modo o programador pode desenvolver um código fluido sem ter que se preocupar em implementar features, pois provavelmente elas já estão prontas para serem usadas. Além disso é de graça, fonte aberta e tem uma comunidade ativa.

**Como é o código?**

Em um website tradicional, a aplicação web espera pela requisição HTTP do browser(ou do cliente). Quando a requisição é recebida a aplicação verificará o método baseado na URL, podendo ser um POST de dados ou GET de dados, por exemplo. Dependendo do que e requerido ele então pode ler informação de um banco de dados ou então escrever informação em um banco de dados, outras requisições com outros tipos de ações podem ser feitas para satisfazer uma requisição. A aplicação então irá retornar uma resposta para o browser, geralmente criando uma pagina HTML para a exibição dos dados lidos ou inseridos neste caso.

As aplicações com Django geralmente agrupam o código que gerencia cada um desses passos em arquivos separados.



* **URLs:** Enquanto é possível processar uma requisição de qualquer URL única a partir de uma função, se torna muito mais sustentável escrever uma função de view para manipular cada recurso. Uma URL mapper é usada para redirecionar requisições HTTP para a view apropriada baseado na própria requisição da URL. A URL mapper pode também combinar certos padrões de strings ou dígitos que aparecem na URL e passar eles para a função de uma view como dados.
* **View**: Uma view é uma função manipuladora de requisições, que recebe requisições HTTP e retorna respostas HTTP. Views acessam os dados necessários para satisfazer uma requisição através das models.
* **Models:** Models, são objetos do Python que definem a estrutura dos dados de uma aplicação e fornecem mecanismos para a sua gerencia( adicionar, modoficar,deletar) e registros de query na base de dados.
* **Templates:** Um templates é um arquivo de texto que define a estrutura ou layout de um arquivo(como uma pagina HTML). Uma view pode dinamicamente criar uma pagina HTML usando um template de HTML, populando isso com dados de um model. Um template pode ser usado para definir a estrutura de qualquer tipo de arquivo, não necessariamente precisa ser um HTML.

**Django Crud - Operações e Desenvolvimento**

CRUD significa Create, Read, Update e Delete. Estes são as quatro operações básicas executadas dentro dos Models. Mostraremos como criar uma aplicação web capaz de fazer essas operações.

1. **Operação Read –** Capacidade da aplicação ler dados da base de dados.
2. **Operação Create –** Capacidade de armazenar dados na base de dados.
3. **Operação Update** – Capacidade da aplicação de editar a um valor já armazenado na base de dados.
4. **Operação Delete** – Capacidade da aplicação de excluir um valor já armazenado na base de dados.

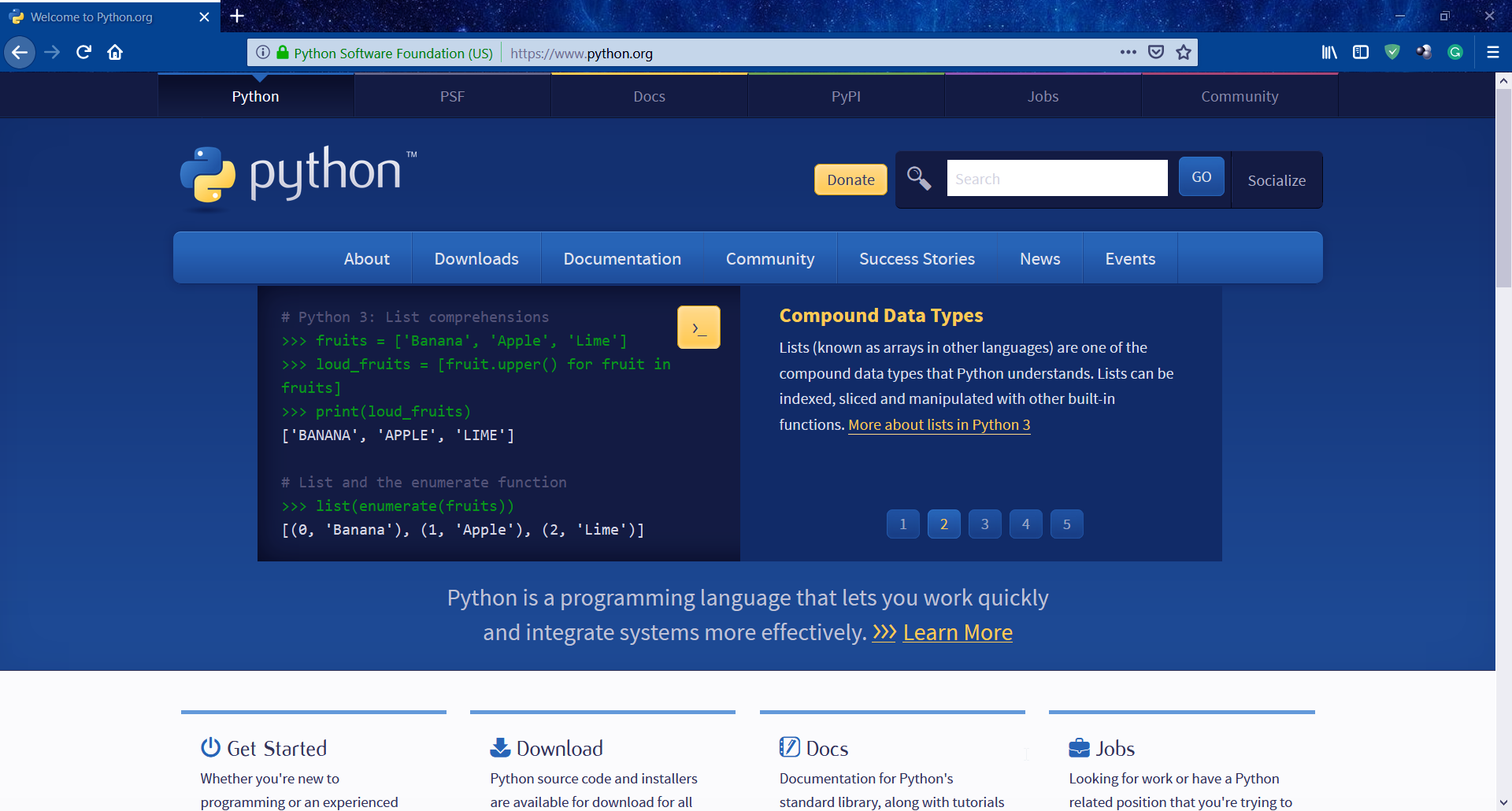
**Django Instalação e configuração do ambiente**

**Passo 1.**

Intalando **Python 3**

Para rodar o Django Framework você precisará do python 3 instalado em seu sistema. É preciso fazer o download do pacote no site official, [www.python.org](http://www.python.org). Instale a versão 3 do Python não a 2.

Quando instanlando o Python 3 não esqueça de assinalar a opção “Add python 3 to my path”.

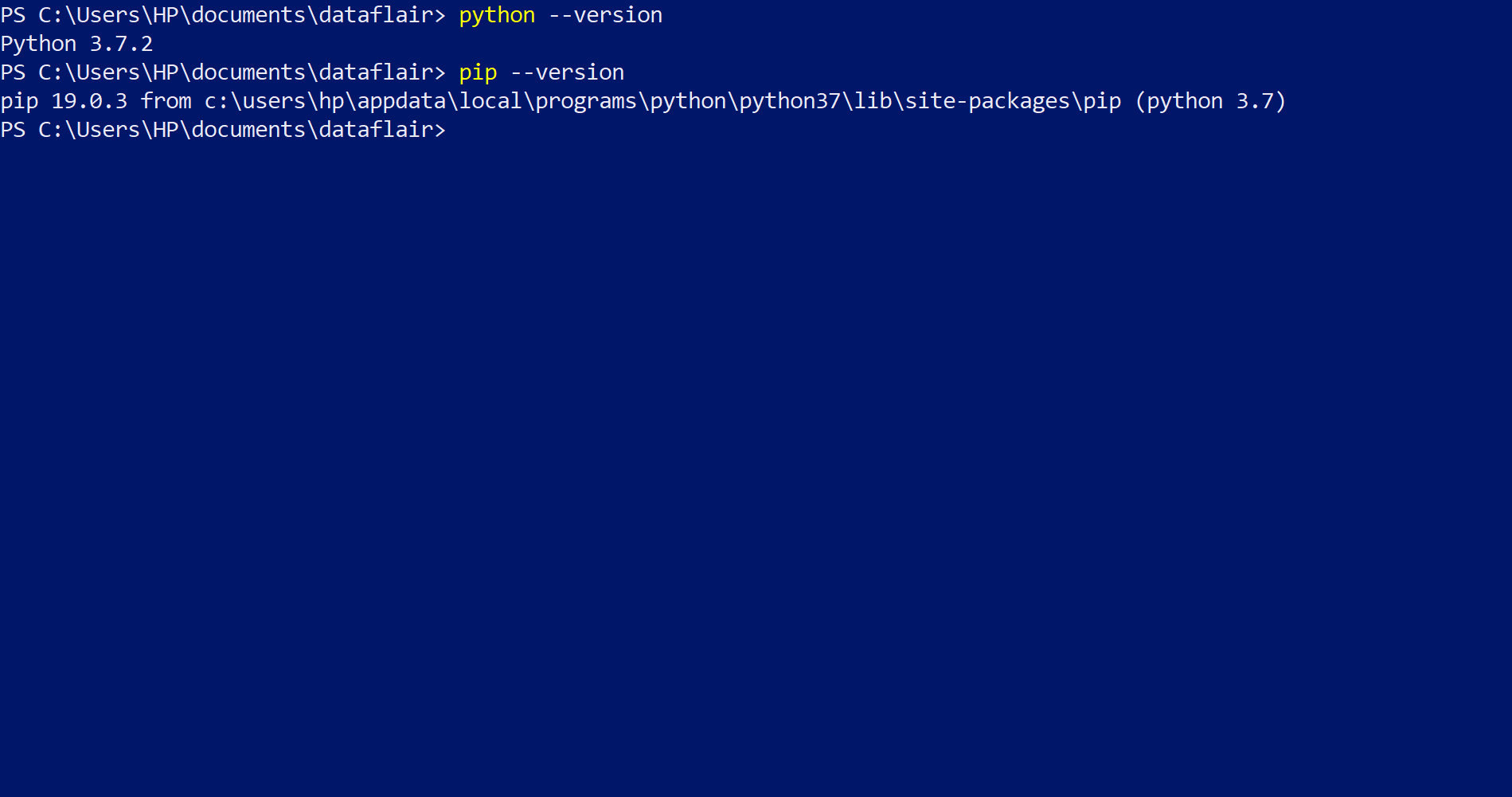


Isso deve também instalar o pip, que é o gerenciador de pacotes do Python e é contantemente usado para o desenvolvimento.

**Para usuario do Windows PowerShell**

Você pode checkar a versão do Python em seu sistema digitando os seguintes comandos no PowerShell:

1. python –version
2. pip –version



**Para usuarios do Ubunto**

1. **$ sudo apt-get update && sudo apt-get -y upgrade** ( Atualizará seu repositório apt)
2. **$ sudo apt-get install python3** (instalara o python3)
3. **$ python3 –V** (verificará se o python foi instalado com sucesso)
4. **$ sudo apt-get install -y python3 pip** (instala o pip)
5. **$ pip3 –V** (verifica se o pip foi instalado com sucesso)

**Passo 2**

**Instalando Virtualenv**

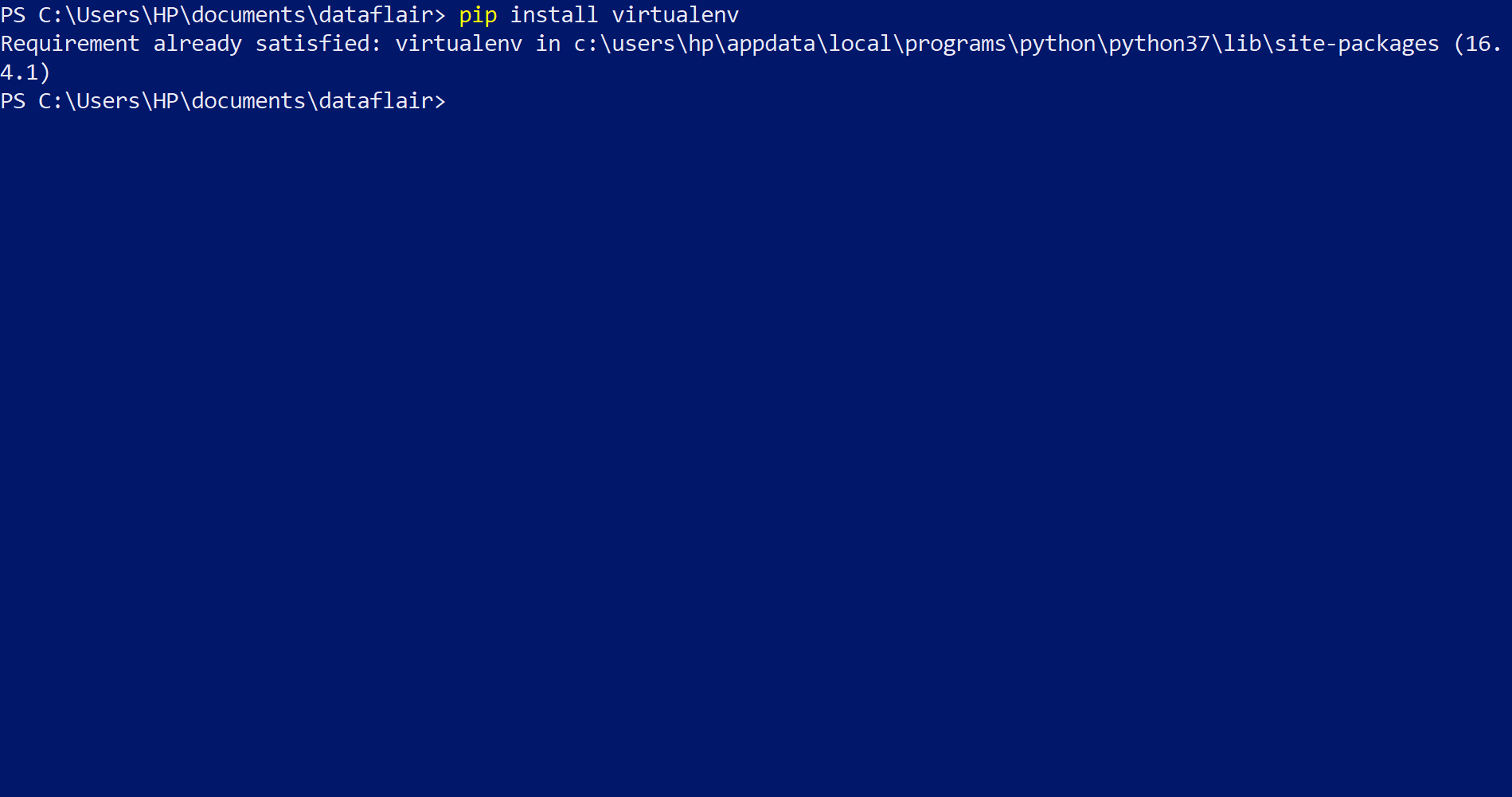
Virtualenv é um pacote do Python que permite a criação de diferentes ambientes virtuais para múltiplos projetos que requerem diferentes versões de software.

Sua principal função é isolar o ambiente do seu projeto, assim qualquer instalação feita dentro do virtualenv não afetara seu sistema.

**Para usuários do Windows PowerShell.**

1. pip install virtualenv

Como mencionado antes pip é o nosso gerenciador de pacotes, portanto usaremos ele para a instalçao do virtualenv



**Para usuários do Ubunto.**

1. **$ pip3 install virtualenv** (instala o virtualenv)
2. **$ virtualenv –-version** (verifica a versão)

**Passo 3**

**Instalando o Django.**

**Para usuarios do WindowsPowerShell**

1. pip install django

**Para usuarios do Ubunto**

1. $ pip install -e django

**Passo 4**

**Instalando Python IDE**

Este passo se restringe mais por conta do desenvolvedor, existem muitas IDE’s e é uma questão muita mais de gosto, porém deixarei dois links como recomendação

* <https://code.visualstudio.com/download>
* <https://www.jetbrains.com/pycharm/download/#section=windows>

**Passo 5**

**Instalando Xampp**

Usaremos base de dados para fazer a aplicação mais interativa e funcional. A base de dados padrão do Djando é o SQLite mas vamos ultilizar SQL. Novamente você pode utilizar qualquer base de dados que voçê se sente mais familiarizado, se preferia pode ultilizar um gerenciador de banco como o Mysql Workbench.

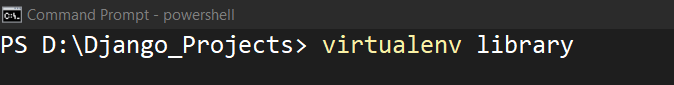
* <https://www.apachefriends.org/download.html> (link para xammp download)

**Passso 6**

**Criando Projeto**

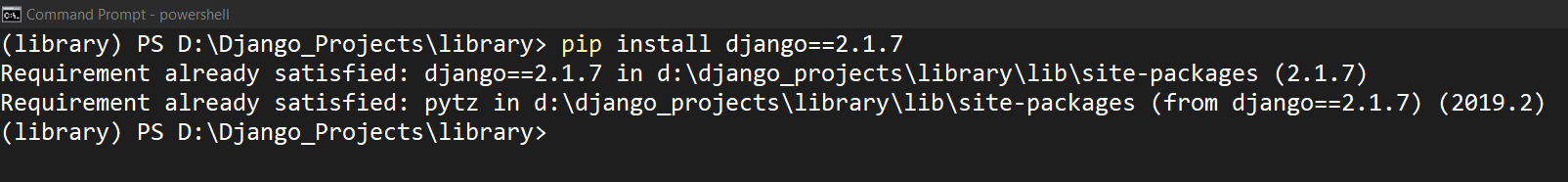
**Para usuarios do Windows PowerShell**

1. virtualenv your\_project\_name



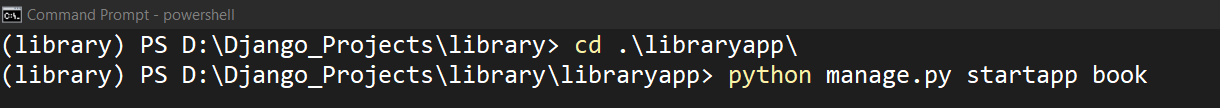
Agora instale o django dentro do ambiente virtual(virtualenv)

1. **Pip install django==2.1.7**



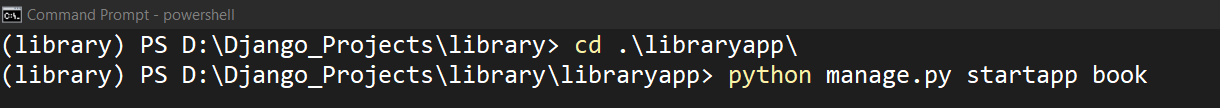
Criando um novo projeto

1. ***django-admin startapp project\_name***



O projeto esta criado pelo django. Agora é so fazer uma nova aplicação. Va para o diretório do arquivo manage.py e faça uma nova apliação.

1. ***python manage.py startapp project\_name***



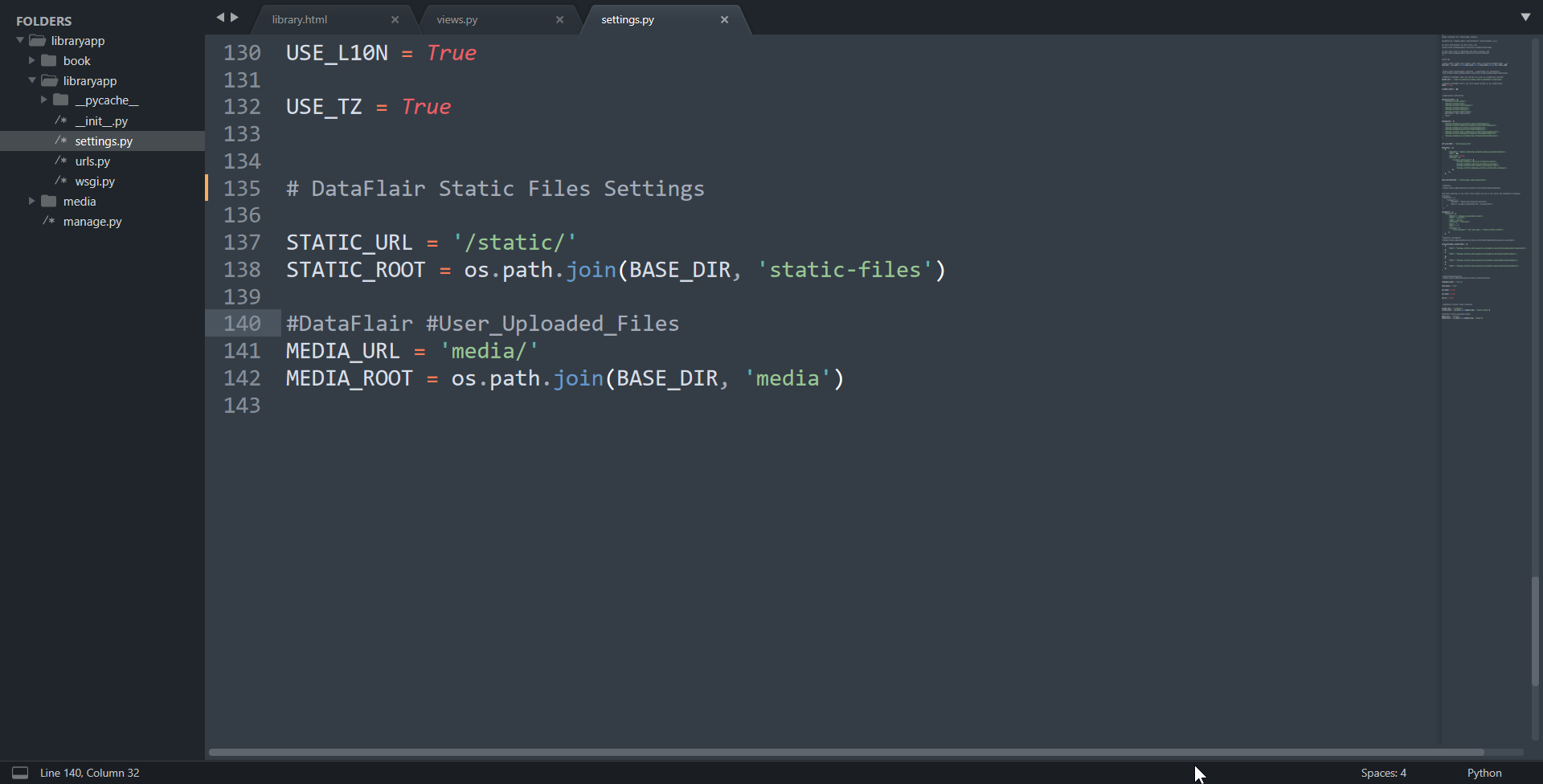
1. **Vá até o arquivo settings.py e adicione o nome da aplicação**



Após conlcuir este passo precisamos estabelecer as configurações dos arquivos estaticos, copie as configurações abaixo e substitua pelas configurações já existentes em seu projeto.

**Codigo:**

1. **# DataFlair Static Files Settings**
2. **STATIC\_URL = '/static/'**
3. **STATIC\_ROOT = os.path.join(BASE\_DIR, 'static-files')**
4. **#DataFlair #User\_Uploaded\_Files**
5. **MEDIA\_URL = 'media/'**
6. **MEDIA\_ROOT=os.path.join(BASE\_DIR, 'media')**



Estas configurações são importantes para armazenamento e para servir arquivos estaticos para o client.

Precisamos também instalar o pacote pillow do Python.

***pip install pillow***

pip install pillow - Django CRUD

Pillow é um pacote do Python que lida com diferentes arquivos de imagem. Nós não usarems pillow diretamente mas Django irá. Portanto precisamos da biblioteca.

**6 Fazendo os Models da nossa Aplicação Agenda**

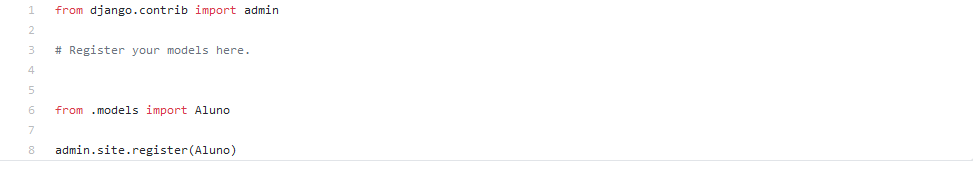


**Entendendo o Codigo:**

Django ira criar três tabelas em um base de dados chamadas Turma, Aluno e Professor, cada tabela terá os campos definidos acima.

**7 Registrando Model no Django Admin**

* admin.site.register(Aluno)



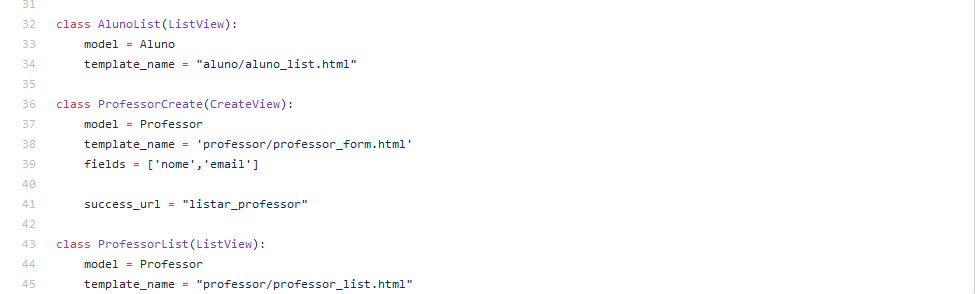
Certo, agora para Impementar isso tudo precisamos rodar alguns comandos



**8 Fazendo as funções da view**

Estaremos editando o arquivo views.py, as funções da view são as operações CRUD no Django.

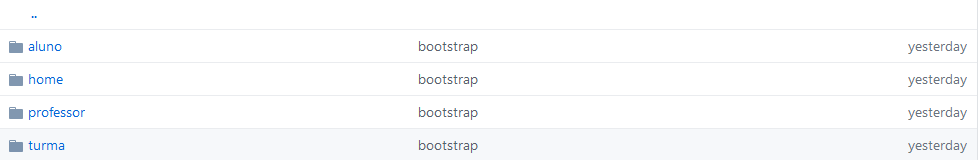


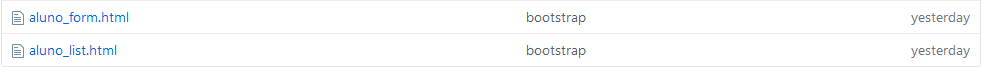


**9. Fazendo Templates**

Abra a pasta templates e crie uma pasta para os templates de alunos, professores e turma( se preferir pode adicionar um template home).

Dentro das pastas dos templates criados iremos adicionar nosso html.





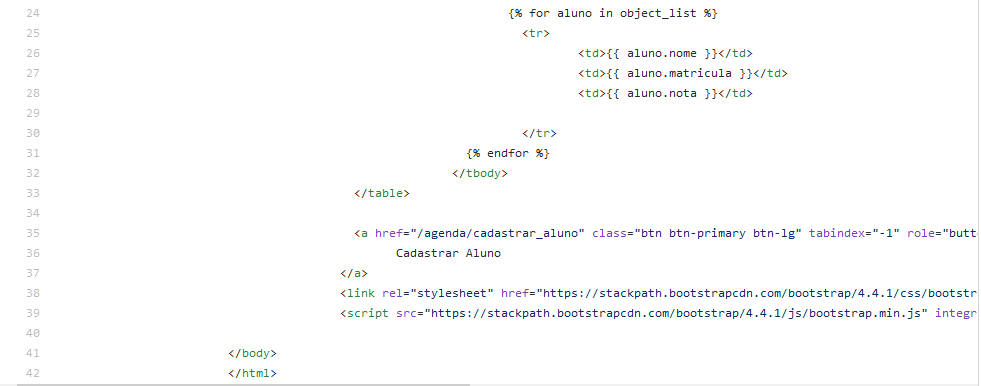
Dentro de cada pasta temos 2 templates, um com um form para armazenar dados no banco e outro para listalos.

1. **Form para armazenamento**

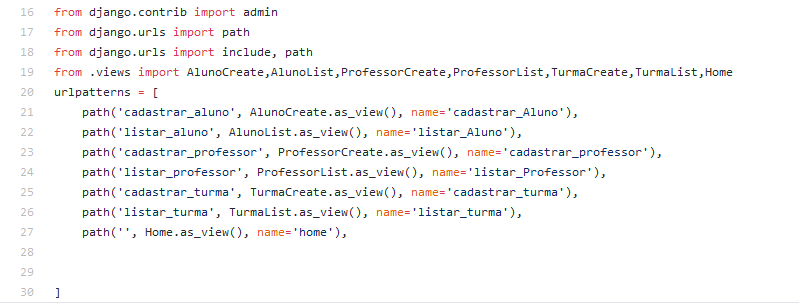


1. **Listagem**





**10. Configurando Urls**



1. **Agora, para rodar seu site e testar se está tudo funcionando corretamente**

* ***python manage.py runserver***
* ***Abra a URL http://127.0.0.1:8000/***